REMARKS

I. INTRODUCTION

Claims 1-7 remain pending in the present application. In view of the following remarks, it is respectfully submitted that all of the presently pending claims are allowable.

II. THE 35 U.S.C. § 102(e) REJECTION SHOULD BE WITHDRAWN

Claims 1-7 stand rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,308,255 to Gorishek IV et al. ("Gorishek"). (See 12/12/07 Office Action, p. 3, lines 16-17).

Claim 1 recites, a "method, comprising: determining a current processing mode of an executing software function; when the current processing mode is a privileged processing mode, executing *a direct program flow control instruction to directly access an instruction within a software having the privileged processing mode*; and when the current processing mode is an unprivileged processing mode, executing *an indirect program flow control instruction to cause execution of the instruction within the software having the privileged processing mode*." (Emphasis added).

Gorishek generally relates to a computer system including a host processor and an emulation coprocessor. (See Gorishek, Abstract). According to Gorishek, the "host processor" comprises hardware configured to execute instructions defined by a host instruction set architecture, while the "emulation coprocessor" comprises hardware configured to execute instructions defined by a different (or foreign) instruction set architecture. (See Id., col. 6, lines 6-28). Accordingly, the host processor executes operating system code as well as application programs that are coded in the host instruction set architecture, while the emulation coprocessor executes the foreign application program. (See Id.). Therefore, when a user submits a command to initiate an application program, the system examines the file format of the application program in order to determine what type of code is included in the application program. (See Id., col. 13, lines 44-49). If *the application program is determined to be coded according to the host instruction set architecture*, the system establishes the process as a normal host process and the application is executed by the host processor. (See Id., col. 13, lines 44-49). Alternatively, if *the application program is determined to be coded according to a foreign instruction set architecture executable by the emulation coprocessor*, then the system invokes the emulation coprocessor interface code in order to initiate the foreign application program. (See Id., col. 13, line 55 – col. 14, line 2). Thus, the initiation of an application program according to the system and method disclosed by Gorishek is dependent on the application program, specifically the code within the application program. In other words, the system according to Gorishek determines which of the two processors (the host or the emulation coprocessor) will execute the code of an application program depending on the format of the code. However, the system according to Gorishek fails to teach or suggest that *a particular instruction within a software having the privileged processing mode* may be executed via a direct program flow control instruction (when the processing mode is privileged) ***and*** may also be executed via an indirect program flow control instruction (when the processing mode is unprivileged), as recited in claim 1. In fact, since the application program described in Gorishek is executed by the host processor ***or*** the emulation coprocessor, Gorishek teaches away from recitations of claim 1.

In the Response to Arguments, the Examiner asserts that the current processing mode by Gorishek is determining examining the file format of the application program, and the system will execute a normal host process ***if*** the file format is in accord with the host instruction set. (See 12/12/07 Office Action, p. 5, lines 6-20). The Examiner continues to state that the system of Gorishek would otherwise execute a foreign application program via the thunk/emulation coprocessor. (See Id.). While the Applicant does not concede that Gorishek discloses "determining a current processing mode of an executing software function," it should be noted the that the Examiner fails to demonstrate that Gorishek discloses an instruction within a software having privileged processing mode is executable by ***both*** a direct program flow control instruction ***and*** an indirect program flow control instruction, as recited in claim 1. Seeing as the system and method disclosed by Gorishek is dependent on the code of the application program requested by the user, an application program that includes code formatted for the host processor can ***only*** be executed by the host processor. Likewise, an application program that includes code according to its foreign instruction set architecture can ***only*** be executed by the emulation coprocessor. Accordingly, a specific application program is executable by ***either*** the host processor ***or*** the emulation coprocessor, but not executable in two different modes. Therefore, as discussed above, the host processor of the Gorishek system is configured to execute one set of instructions while the emulation coprocessor is configured to execute a different set of instructions. (See Gorishek, col. 6, lines 16-28).

In contrast to Gorishek, the method recited in claim 1 of the present invention is not dependent on the file format of an instruction within a software. However, the manner in which the instructions recited in claim 1 are executed is dependent on the current processing mode. It is clear within the claim language that an instruction within the software may be executed directly when the processing mode is privileged, and the same instruction within the software may be executed indirectly when the processing mode is unprivileged. Accordingly, the instructions within the software, described in claim 1, may be executed in both processing modes, regardless of the file format of the instructions. The same cannot be said for the system according to Gorishek. As discussed above, and as reiterated by the Examiner, Gorishek examines the file format in order to have the application program executed by the host processor or executed by the emulation coprocessor. (See 12/12/07 Office Action, p. 5, lines 16-20).

Thus, it is respectfully submitted that Gorishek does not disclose nor suggest, a "method, comprising: determining a current processing mode of an executing software function; when the current processing mode is a privileged processing mode, executing *a direct program flow control instruction to directly access an instruction within a software having the privileged processing mode*; and when the current processing mode is an unprivileged processing mode, executing *an indirect program flow control instruction to cause execution of the instruction within the software having the privileged processing mode*. as recited in claim 1. Accordingly, Applicants respectfully submit that for at least the reasons stated above, claim 1 of the present application is not anticipated by Gorishek, and request that the rejection of this claim be withdrawn. As claims 2-5 depend from, and therefore include all the limitations of claim 1, it is hereby submitted that these claims are also allowable.

Claim 6 recites, a "method, comprising: identifying a program code segment implementing an access to a memory area to be executed within a privileged processing mode; replacing the program code segment with a substitute code segment; wherein the substitute code segment includes program code to identify a current processing mode of the program code segment, execute a direct program flow control instruction if the current processing mode is the privileged processing mode, and execute an indirect program flow control instruction if the current processing mode is an unprivileged processing mode." Thus, for the reasons described above with reference to claim 1, it is respectfully submitted that claim 6 is also allowable.

Claim 7 recites, a "computer readable medium encoded with a software application, comprising: a software code implementing application functionality; and a smart system call into an operating system; wherein the smart system call comprises the software code to identify a current processing mode of a program code segment, execute a direct program flow control instruction if the current processing mode is a privileged processing mode, and execute an indirect program flow control instruction if the current processing mode is an unprivileged processing mode." Thus, for the reasons described above with reference to claim 1, it is respectfully submitted that claim 7 is also allowable.

CONCLUSION

In light of the foregoing, Applicants respectfully submit that all of the now pending claims are in condition for allowance. All issues raised by the Examiner having been addressed. An early and favorable action on the merits is earnestly solicited.

Respectfully submitted,

Dated: February 12, 2008

By:________________
Michael J. Marcin (Reg. No. 48,198)

Fay Kaplun & Marcin, LLP
150 Broadway, Suite 702
New York, NY 10038
Phone: 212-619-6000
Fax: 212-619-0276